AFOSR-TR 97-0422

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	23 May 97	Final: 4/1/93-3/31/97

4. TITLE AND SUBTITLE
Semantic Theories and Automated Tools for Real-Time and Probabilistic Concurrent Systems

5. FUNDING NUMBERS
AFOSR grant no. F49620-93-1-0250

6. AUTHOR(S)
Scott A. Smolka

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Department of Computer Science
SUNY at Stony Brook
Stony Brook, NY 11794-4400

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION UNLIMITED

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The main objective of this project was to develop new semantic theories and automated tools for real-time and probabilistic concurrent systems; that is, systems of coordinating processes that exhibit behavior of a probabilistic or statistical nature and which must meet real-time constraints. The main results achieved include a new semantic framework for reasoning about the relative reliability of probabilistic systems in different operating environments; an efficient algorithm for checking whether a specification of a real-time concurrent system satisfies a correctness property specified in a real-time temporal logic; and a new model of soft real-time systems that allows users to make rigorous statements about the likelihood with which systems are guaranteed to meet deadlines. A number of these results have been incorporated into the Concurrency Factory verification toolkit. In turn, the Factory has provided a platform for technology transfer with several Long Island companies, including Parker-Hannifin, Reuters America, and Northrop Grumman.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Real-time systems, probabilistic processes, computer-aided verification, model checking, communication protocols	7
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
unclassified	unclassified	unclassified	UL

AFOSR F49620-93-1-0250

Semantic Theories and Automated Tools for Real-Time and Probabilistic Concurrent Systems

Scott A. Smolka
Department of Computer Science
SUNY at Stony Brook

Final Technical Report

Introduction

The main objectives of AFOSR Grant no. F49620-93-1-0250 were the following:

- To develop new semantics theories for real-time and probabilistic concurrent systems; that is, systems that exhibit behavior of a probabilistic or statistical nature and which must meet real-time constraints.
- To embed our new theories in the Concurrency Factory, an integrated toolkit for the specification, simulation, automatic verification, and implementation of concurrent systems.
- To apply our semantic theories to real-life systems, such as communication protocols, embedded systems, process control systems, and digital control units.
- To perform a technology transfer of our research results to industry and DoD entities.

The research supported by AFOSR Grant no. F49620-93-1-0250 produced the following results, which represent the main accomplishments under the grant:

Testing Preorders for Probabilistic Processes A new semantic framework was developed for reasoning about the relative reliability of concurrent processes in different operating environments.

Local Model Checking for Real-Time Systems An efficient algorithm was invented for checking whether a specification of a real-time concurrent system satisfies correctness properties specified in a real-time temporal logic.

Probabilistic Input/Output Automata Probabilistic Input/Output Automata is a new model of probabilistic concurrent computation that facilitates the analysis of delay and probability in distributed systems.

New Semantic Model for Soft Real-Time Systems A new semantic theory was developed that permits the modeling of probabilistic real-time systems and allows users to make rigorous statements about the likelihood with which systems are guaranteed to meet deadlines.

The Concurrency Factory CASE Environment The Concurrency Factory is an integrated environment for specification, simulation, verification, and implementation of concurrent systems. It embodies a number of the concepts developed during the grant period.

Below, these results are described in greater detail.

Testing Preorders for Probabilistic Processes

Communicating systems often exhibit behavior that is probabilistic or statistical in nature. For example, one may observe that a faulty communication link drops a message 2% of the time or that a site in a network is down with probability 0.052. It is therefore interesting to consider *probabilistic processes* as a system specification method. In a probabilistic process, nondeterministic choice points are augmented with probability information in the form of distributions on outgoing transitions.

In [CSZ92, YCDS94] a *testing preorder* is presented that relates probabilistic processes in terms of their relative reliability in different operating environments. In this setup, environments are modeled by *tests*, which are themselves probabilistic processes equipped with a set of *success* states. The probabilistic testing preorder is based on the notion of the probability by which a process passes a test. Process $\mathcal{P}$ is less than process $\mathcal{Q}$ (denoted $\mathcal{P} \sqsubseteq \mathcal{Q}$) in the preorder when, for all tests $\mathcal{T}$, the probability that $\mathcal{P}$ passes $\mathcal{T}$ is no greater than the probability that $\mathcal{Q}$ passes $\mathcal{T}$.

We have shown that the probabilistic testing preorder enjoys close connections with the classical testing theory of De Nicola and Hennessy [DNH83, Hen88] for nondeterministic processes. In particular, if two probabilistic processes are related by the probabilistic testing preorder, then their "deprobabilized" images (obtained by erasing the probabilities in the underlying transition systems) are related by the testing preorder of De Nicola and Hennessy. Thus, the testing theory of probabilistic processes is a refinement of the testing theory of nonprobabilistic processes.

While intuitively appealing, the operational definition of $\sqsubseteq$ can be difficult to reason about. In particular, establishing that one process is related to another requires a consideration of the behavior of both processes in the context of all possible tests. To circumvent this obstacle, we have also developed an alternative, more denotational characterization of $\sqsubseteq$ to ease the task of establishing relationships between probabilistic processes. Moreover, this characterization is *fully abstract*, relating probabilistic processes in exactly the same manner as $\sqsubseteq$.

Local Model Checking for Real-Time Systems

Model checking is the problem of determining whether a system specification satisfies, i.e. is a model of, a correctness property specified as a formula in some temporal logic. In the *local* approach to model checking, the whole state space of the system under investigation need not be explored, but rather only that portion necessary to determine the truth or falsehood of the logical formula.

As reported in [SS95], the problem of extending local model checking to real-time specifications has been investigated. The main result of this investigation was a local algorithm for model checking in a real-time extension of the alternation-free modal mu-calculus. The principal innovations of the algorithm, called TMC (Timed Model Checking), are the following:

- TMC is, to our knowledge, the first local model checking algorithm to be proposed for the verification of real-time systems. Thus, like its counterparts for untimed systems, verification is carried out in a goal-directed manner and only those portions of the state space necessary to determine the truthhood of the formula are explored.
- The temporal logic used by TMC represents the first true extension of the modal mu-calculus to real-time systems. This logic supports all of the original operators of the modal mu-calculus as well as the two new "time modalities" of Holmer et al: necessity and possibility of "time successors". Moreover, this logic achieves a clear separation of the time-dependent aspects from the untimed ones. This resulted in the reuse of a significant portion of the code of the Concurrency Factory's local model checker for the modal mu-calculus (discussed below) when implementing the local model checker for the real-time logic.
- Like most algorithms dealing with real-time systems, TMC works with a finite quotient of the statespace as the statespace itself is inherently infinite. For maximal efficiency, TMC obtains a quotient that is *as coarse as possible* in the following sense: refinements of the quotient are carried out only when necessary to satisfy "clock constraints" in the logical formula or timed automaton used to represent the system under investigation. In this sense, our data structures are optimal with respect to the given formula and automaton.

TMC performs model checking by constructing a data structure representing the "product" of the given logical formula and the transition system induced by the given timed automaton. Each node of this "region product graph" (RPG) represents the value of a logical variable for some set of "timed states", or region. The RPG is constructed on-the-fly and explored in a depth-first manner, until nodes with a known value are found. After each step of the RPG construction, partitioning (or splitting) of nodes may be necessary to achieve stability with respect to the relevant clock constraints. Initial experimental results have shown TMC to be highly competitive efficiency-wise with existing real-time model checkers.

Probabilistic Input/Output Automata

Probabilistic I/O automata[WSS94, WSS97] is a framework we developed for reasoning algebraically about asynchronous, probabilistic and real-time systems. Probabilistic I/O automata are based on the *I/O automaton* model of Lynch and Tuttle [LT87]. In an I/O automaton, a distinction is made between *input* actions, which come from the environment and are always enabled, and *output* actions, which are locally controlled by the automaton itself. Accordingly, in probabilistic I/O automata, separate probability distributions are associated with each input action and a single distribution with all locally controlled actions. As such, no relative probability is specified between different inputs (or between inputs and locally controlled actions) since these choices are under the control of the environment.

Each state of a probabilistic I/O automaton is equipped with a *delay parameter* representing the expected delay before the automaton executes an action from a given state. The delay parameter plays a dual role: it facilitates the definition of asynchronous parallel composition among probabilistic I/O automata, and at the same time introduces a convenient notion of timing into the model. The former is particularly important given that virtually all previous work in the field has focused on synchronously composed probabilistic systems.

We have also defined a *testing equivalence* for probabilistic I/O automata, which, like our work in [CSZ92, YCDS94], is based on the natural notion of an automaton passing a test with a certain probability. Using *probabilistic behavior maps*, we have obtained a fully abstract alternative characterization of the testing equivalence, which eases the task of proving (or disproving) two probabilistic I/O automata testing equivalent. By "fully abstract" we mean that the alternative characterization identifies two automata if and only if they are testing equivalent.

We have used probabilistic I/O automata to model the Boeing 777 Stabilizer Position Indicator (SPI) program. The SPI application is discussed further below. Ongoing work involves the use of probabilistic I/O automata to compositionally, and hence efficiently, determine expected delays between events in distributed systems.

New Semantic Model for Soft Real-Time Systems

In the literature on real-time systems (see, for example, [LS96]), a distinction is often drawn between hard and soft real-time systems. In a *hard real-time system*, all deadlines must be met, as the consequences of failing to meet a deadline can be devastating. Examples of such systems include process control systems for nuclear power plants and fly-by-wire avionics systems. In a *soft real-time system*, the consequences of failing to meet a deadline are not nearly as grave, and thus a certain percentage of missed deadlines can be tolerated. Examples of soft real-time systems include financial data delivery systems and the U.S. Postal Priority Mail system.

During the third year of the grant, a new semantic theory of soft real-time systems was developed. This theory permits the modeling of probabilistic real-time systems and allows users to make rigorous statements about the likelihood with which systems are guaranteed to meet deadlines. In contrast, most existing models only support the analysis of hard real-time systems in which no deadlines can ever be missed.

The key components of the framework are: (1) a specification language PDP (for Probabilistic and Discrete-time Processes) that incorporates both probabilistic and timing aspects of process behavior; (2) a formal operational semantics for PDP given as a recursively defined probability distribution function over process terms and atomic actions; and (3) a natural notion of a process passing a test with a certain probability, where a test is a process with the added capability of reporting success. By encoding deadlines as tests, the probability of a process passing a test may now be interpreted as the probability of the process meeting a deadline, thereby capturing the essence of soft real-time. A simple video frame transmission example was developed to illustrate the approach.

The Concurrency Factory and Industrial Interactions

The Concurrency Factory is an integrated toolset for specification, simulation, verification, and implementation of concurrent systems, such as communication protocols and process control systems. The original source of funding for the Factory is NSF grant CCR-9120995, co-principal investigators Philip Lewis and Scott Smolka (SUNY, Stony Brook), and Rance Cleaveland (N.C. State). AFOSR Grant no. F49620-93-1-0250 has provided key additional support for the project, especially with regard to the Factory's capability to handle real-time specifications and to produce executable code from specifications.

Two themes central to the Concurrency Factory project are the following: the use of *process algebra* [Mil89, BK84, Hoa85] as the underlying formal model, and the provision of *practical* support for process algebra. By "practical" we mean that the Factory should be usable by protocol engineers and software developers who are not necessarily familiar with formal verification, and it should be usable on problems of real-life scale, such as those found in the avionics industry.

The main features of the Concurrency Factory are the following:

- A graphical user interface, VTView/VTSim, that allows the non-expert to design and simulate concurrent systems using process algebra. VTView is a graphical editor for hierarchically structured networks of finite-state processes, and VTSim is a sophisticated environment for the simulation and testing of VTView-constructed specifications.
- A textual user interface based on the language VPL, a simple parallel programming language that provides support for a small collection of data types, such as fixed size integers and integer arrays. VPL makes it easier to specify concurrent programs in which actual data values are passed between processes. Yet, the underlying systems

are still finite-state which means that VPL programs, like VTView specifications, are amenable to automatic verification.

- A suite of design and analysis algorithms that currently includes a local model checker for the alternation-free modal mu-calculus and its real-time extension, and a bisimulation checker based on the algorithm of Kanellakis and Smolka. Care has been taken to ensure that these algorithms are efficient enough to be used on real-life systems.

- A graphical compiler that transforms VTView specifications into executable code. Our current version produces Facile code, a concurrent language that symmetrically integrates many of the features of Standard ML and CCS. Facile programs execute as independent processes communicating over TCP/IP, and are thus truly concurrent/distributed. We are also considering compilation into Ada94. The graphical compiler relieves the user of the burden of manually recoding their designs in the target language of their final system.

The Concurrency Factory is written in C++ and executes under X-Windows, using Motif as the graphics engine, so that it is efficient, easily extendible, and highly portable. It is currently running on SUN SPARCstations under SunOS and Solaris.

With regard to the theories and tools developed under AFOSR Grant no. F49620-93-1-0250, the Concurrency Factory serves as our vehicle for **technology transfer**. Specifically, we have interacted with the following Long Island companies:

- We have used the Concurrency Factory to specify and analyze a highly fault-tolerant communications protocol used by Reuters America in their worldwide financial trading network.

- The Factory was used, along with the related probabilistic I/O automata technology [WSS97], to specify the Stabilizer Position Indicator module of the Boeing 777. The SPI module was designed and constructed by Parker-Hannifin.

- We have used the Concurrency Factory in our recent interactions with Northrop Grumman to specify and verify a number of key properties of the E-2C communications protocol used on AWACS aircraft for reliable communication between mission computers and tactical workstations.

References

[BK84] J. A. Bergstra and J. W. Klop. Process algebra for synchronous communication. *Information and Computation*, 60:109–137, 1984.

[CSZ92] R. Cleaveland, S. A. Smolka, and A. E. Zwarico. Testing preorders for probabilistic processes. In W. Kuich, editor, *Automata, Languages and Programming (ICALP '92)*, volume 623 of *Lecture Notes in Computer Science*, pages 708–719, Vienna, July 1992. Springer-Verlag.

[DNH83] R. De Nicola and M. C. B. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1983.

[Hen88] M. C. B. Hennessy. *Algebraic Theory of Processes.* MIT Press, Boston, Mass., 1988.

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, London, 1985.

[LS96] J. Lee and S. H. Son. Performance of concurrency control algorithms for real-time database systems. In V. Kumar, editor, *Performance of Concurrency Control Mechanisms in Centralized Database Systems*, pages 429–460. Prentice-Hall, 1996.

[LT87] N. A. Lynch and M. Tuttle. Hierarchical correctness proofs for distributed algorithms. In *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing*, pages 137–151, 1987.

[Mil89] R. Milner. *Communication and Concurrency.* International Series in Computer Science. Prentice Hall, 1989.

[SS95] O. Sokolsky and S. A. Smolka. Local model checking for real-time systems. In *Proceedings of the 7th International Conference on Computer-Aided Verification.* American Mathematical Society, 1995.

[WSS94] S.-H. Wu, S. A. Smolka, and E. W. Stark. Compositionality and full abstraction for probabilistic I/O automata. In *Proceedings of CONCUR '94 — Fifth International Conference on Concurrency Theory*, Uppsala, Sweden, August 1994.

[WSS97] S.-H. Wu, S. A. Smolka, and E. W. Stark. Compositionality and full abstraction for probabilistic I/O automata. *Theoretical Computer Science*, 1997. To appear.

[YCDS94] S. Yuen, R. Cleaveland, Z. Dayar, and S. A. Smolka. Fully abstract characterizations of testing preorders for probabilistic processes. In B. Jonsson and J. Parrow, editors, *CONCUR '94*, volume 836 of *Lecture Notes in Computer Science*, Uppsala, Sweden, August 1994. Springer-Verlag.

People Involved in the Project

Ph.D. students Oleg Sokolsky and Sue-Hwey Wu were supported by the grant. Oleg worked primarily on the local model checking algorithm for real-time systems and on the design and implementation of the Concurrency Factory. Sue worked on the development of the probabilistic I/O automata model and its application to the Boeing SPI module. Both Oleg and Sue received their Ph.D.'s in May 1996. Oleg nows works for CCCC in Philadelphia. Their theses were published as:

O. Sokolsky, *Efficient Graph-Based Algorithms for Model Checking in the Modal Mu-Calculus*, Department of Computer Science, SUNY at Stony Brook, April 1996.

S.-H. Wu, *Compositionality and Full Abstraction for Probabilistic I/O Automata*, Department of Computer Science, SUNY at Stony Brook, May 1996.

Stony Brook Computer Science Professor Eugene Stark was also involved in the project. He and I co-advised Sue-Hwey Wu in her Ph.D. studies.

Publications Stemming from the Project

1. S.-H. Wu, S.A. Smolka and E. Stark, "Compositionality and Full Abstraction for Probabilistic I/O Automata." *Theoretical Computer Science.* To appear, 1997.

2. R. Cleaveland and S.A. Smolka, "Strategic Directions in Concurrency Research." *ACM Computing Surveys*, Vol. 28, No. 4, pp. 607-625 (December 1996).

3. Y.S. Ramakrishna and S.A. Smolka, "Partial-Order Reduction in the Weak Modal Mu-Calculus," *Proceedings of the Eight International Conference on Concurrency Theory (CONCUR '97)*, Warsaw, Poland, Lecture Notes in Computer Science, Springer-Verlag (July 1997).

4. R. Cleaveland, I. Lee, P.M. Lewis, and S.A. Smolka, "A Theory of Testing for Soft Real-time Processes," *Proceedings of the Eighth International Conference on Software Engineering and Knowledge Engineering*, Lake Tahoe, Nevada. Published by Knowledge Systems Institute, Skokie, Illinois, ISBN 0-9641699-3-2 (May 1996).

5. S.A. Smolka and B. Steffen, "Priority as Extremal Probability." *Formal Aspects of Computing*, Vol. 8, pp. 585-606 (1996).

6. Y.-J. Joung and S.A. Smolka, "Strong Interaction Fairness via Randomization," *16th IEEE International Conference on Distributed Computing Systems*, Hong Kong, pp./ 475-483 (May 1996).

7. R. Cleaveland, P.M. Lewis, S.A. Smolka, and O. Sokolsky, "The Concurrency Factory: A Development Environment for Concurrent Systems," *Computer Aided Verification '96*, Springer-Verlag (1996).

8. R. Cleaveland, P.M. Lewis, S.A. Smolka, and O. Sokolsky, "The Concurrency Factory Software Development Environment," *TACAS '96*, Springer-Verlag (1996).

9. O. Sokolsky and S.A. Smolka, "Local Model Checking for Real-Time Systems," *Computer-Aided Verification '95*, American Mathematical Society (July 1995).

10. S. Zhang, S.A. Smolka, and O. Sokolsky, "On the Parallel Complexity of Bisimulation and Model Checking," in *Modal Logic and Process Algebra: A Bisimulation Perspective*, CSLI Lecture Notes No. 53, pp. 257-288 (1995).

11. R. van Glabbeek, S.A. Smolka, and B.U. Steffen, "Reactive, Generative, and Stratified Models of Probabilistic Processes." *Information and Computation*, Vol. 121, No. 1, pp. 59-80 (August 1995).

12. J.C.M. Baeten, J.A. Bergstra, and S.A. Smolka, "Axiomatizing Probabilistic Processes: ACP with Generative Probabilities." *Information and Computation*, Vol. 121, No. 2, pp. 234-255 (September 1995).

13. S. Zhang, S.A. Smolka, and O. Sokolsky, "On the Parallel Complexity of Model Checking in the Modal Mu-Calculus," *Proceedings of Ninth Annual IEEE Symposium on Logic in Computer Science*, pp. 154-163, London, England (July 1994).

14. R. Cleaveland, J.N. Gada, P.M. Lewis, S.A. Smolka, O. Sokolsky, and S. Zhang, "The Concurrency Factory — Practical Tools for Specification, Simulation, Verification, and Implementation of Concurrent Systems," *Proceedings of DIMACS Workshop on Specification of Parallel Algorithms*, Princeton, NJ (May 1994).

15. R. Gupta, S. Bhaskar, and S.A. Smolka, "On Randomization in Sequential and Distributed Algorithms," *ACM Computing Surveys*, Vol. 26, No. 1, pp. 7-86 (March 1994).

16. A. Uselton and S.A. Smolka, "A Compositional Semantics for Statecharts using Labeled Transition Systems," *Proceedings of CONCUR '94 — Fifth International Conference on Concurrency Theory*, Uppsala, Sweden (Aug. 1994).

17. S.-H. Wu, S. A. Smolka and E. Stark, "Compositionality and Full Abstraction for Probabilistic I/O Automata," *Proceedings of CONCUR '94 — Fifth International Conference on Concurrency Theory*, Uppsala, Sweden (Aug. 1994).

18. S. Yuen, R. Cleaveland, Z. Dayar, and S.A. Smolka, "Fully Abstract Characterizations of Testing Preorders for Probabilistic Processes," *Proceedings of CONCUR '94 — Fifth International Conference on Concurrency Theory*, Uppsala, Sweden (Aug. 1994).

19. A. Uselton and S.A. Smolka, "A Process-Algebraic Semantics for Statecharts via State Refinement," *Proceedings of PROCOMET '94*, San Miniato, Italy. Proceedings published by Elsevier/North Holland (1994).

20. O. Sokolsky and S.A. Smolka, "Incremental Model Checking in the Modal Mu-Calculus," *Computer-Aided Verification '94*, American Mathematical Society (June 1994).

21. S. Zhang and S.A. Smolka, "Efficient Parallelization of Equivalence Checking Algorithms," *Proceedings of FORTE '92 - Fifth International Conference on Formal Description Techniques*, M. Diaz and R. Groz (eds.), pp. 133–146 (Oct. 1992).

22. J.C.M. Baeten, J.A. Bergstra, and S.A. Smolka, "Axiomatizing Probabilistic Processes: ACP with Generative Probabilities," *Proceedings of CONCUR '92 - Third International Conference on Concurrency Theory*, Stony Brook, NY (Aug. 1992). Proceedings published as *Lecture Notes in Computer Science*, Vol. 630, W.R. Cleaveland (ed.), Springer-Verlag (1992).

23. R. Cleaveland, S.A. Smolka, and A. Zwarico, "Testing Preorders for Probabilistic Processes," *Proceedings of 19th International Colloquium on Automata, Languages and Programming*, Vienna, Austria (July 1992). Proceedings published as *Lecture Notes in Computer Science*, Vol. 623, W. Kuich (ed.), Springer-Verlag (1992).